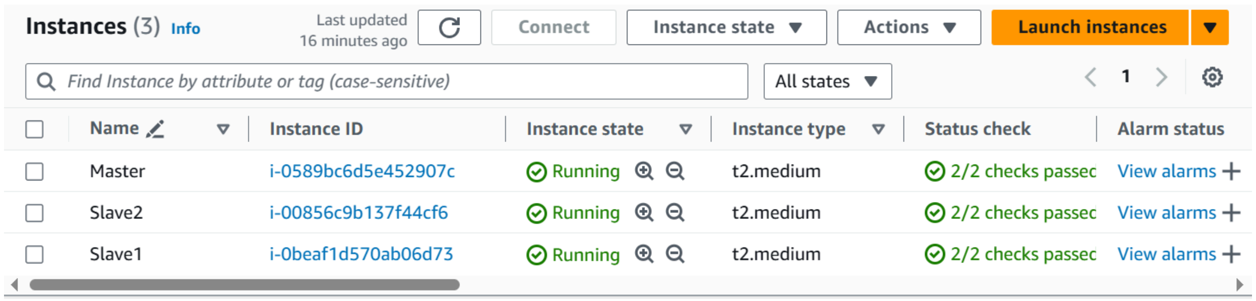
Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Lab Exp 03

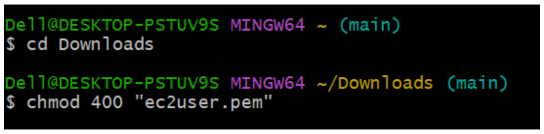
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up aKubernetes Cluster on Linux Machines/Cloud

Step 1: Create 3 EC2 instances (1 master and 2 slaves). Select SSH option in the inbound rules. Created a key pair to be used commonly between all 3 instances created.
I selected AWS Linux as my operating system and enabled t2 medium option for kubernetes cluster to run smoothly

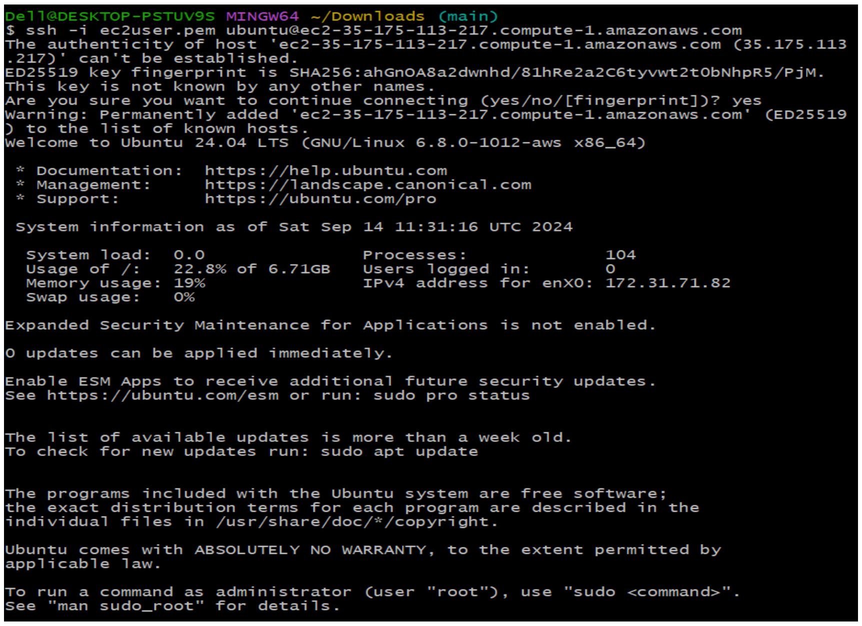

Instances (3) Info — Last updated 16 minutes ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
Master	i-0589bc6d5e452907c	Running	t2.medium	2/2 checks passec	View alarms +
Slave2	i-00856c9b137f44cf6	Running	t2.medium	2/2 checks passec	View alarms +
Slave1	i-0beaf1d570ab06d73	Running	t2.medium	2/2 checks passec	View alarms +

Step 2: Open git bash. Change your directory to Downloads and run chmod command on the key pair file that we created for all the EC2 instances that we launched in the earlier step. Hers, i have created a key pair with the name "ec2user" and gave it an extension .pem.
After assigning the key pair file the appropriate permissions, run the ssh command in the following format:
ssh -i <key pair.pem> ubuntu@<public ip address of the instance(DNS)>
Perform this command on all the 3 instances



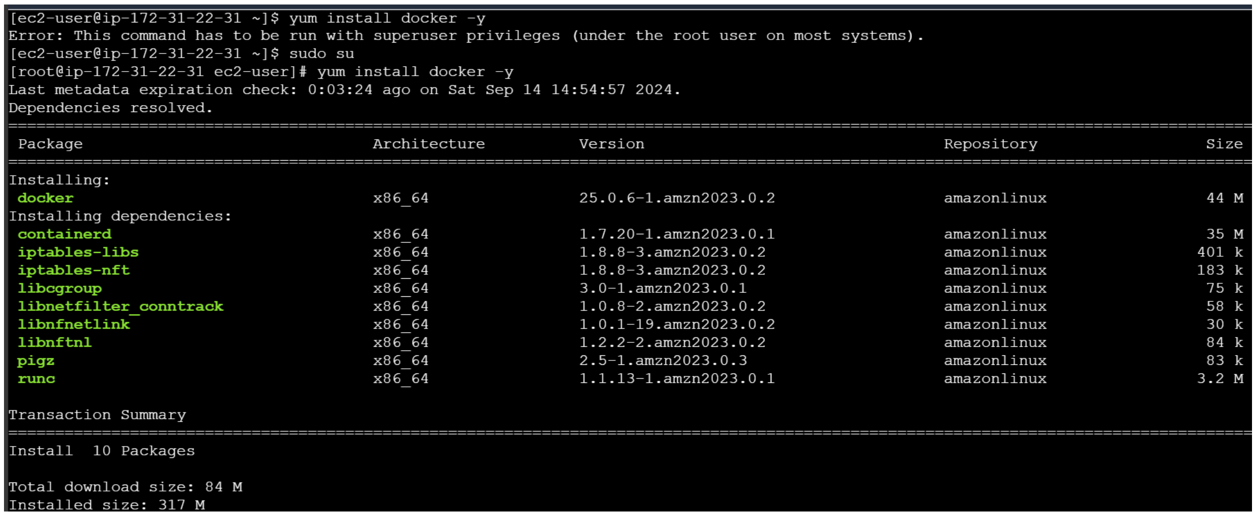
```
Dell@DESKTOP-PSTUV9S MINGW64 ~ (main)
$ cd Downloads

Dell@DESKTOP-PSTUV9S MINGW64 ~/Downloads (main)
$ chmod 400 "ec2user.pem"
```



```
Dell@DESKTOP-PSTUV9S MINGW64 ~/Downloads (main)
$ ssh -i ec2user.pem ubuntu@ec2-35-175-113-217.compute-1.amazonaws.com
The authenticity of host 'ec2-35-175-113-217.compute-1.amazonaws.com (35.175.113
.217)' can't be established.
ED25519 key fingerprint is SHA256:ahGnOA8a2dwnhd/81hRe2a2C6tyvwt2tObNhpR5/PjM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-175-113-217.compute-1.amazonaws.com' (ED25519
) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Sep 14 11:31:16 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 19%               IPv4 address for enX0: 172.31.71.82
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Name: Mohit Kerkar Div: D15C Roll No: 23

Step 3: Docker Installation
Perform this step on all 3 instances



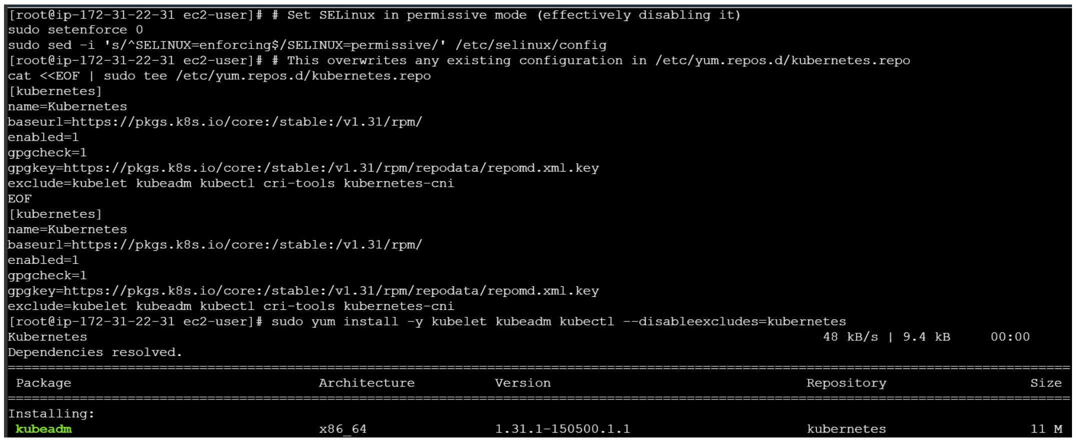
```
[ec2-user@ip-172-31-22-31 ~]$ yum install docker -y
Error: This command has to be run with superuser privileges (under the root user on most systems).
[ec2-user@ip-172-31-22-31 ~]$ sudo su
[root@ip-172-31-22-31 ec2-user]# yum install docker -y
Last metadata expiration check: 0:03:24 ago on Sat Sep 14 14:54:57 2024.
Dependencies resolved.
=====================================================================================================================================
 Package                       Architecture         Version                              Repository                 Size
=====================================================================================================================================
Installing:
 docker                        x86_64               25.0.6-1.amzn2023.0.2                amazonlinux                44 M
Installing dependencies:
 containerd                    x86_64               1.7.20-1.amzn2023.0.1                amazonlinux                35 M
 iptables-libs                 x86_64               1.8.8-3.amzn2023.0.2                 amazonlinux               401 k
 iptables-nft                  x86_64               1.8.8-3.amzn2023.0.2                 amazonlinux               183 k
 libcgroup                     x86_64               3.0-1.amzn2023.0.1                   amazonlinux                75 k
 libnetfilter_conntrack        x86_64               1.0.8-2.amzn2023.0.2                 amazonlinux                58 k
 libnfnetlink                  x86_64               1.0.1-19.amzn2023.0.2                amazonlinux                30 k
 libnftnl                      x86_64               1.2.2-2.amzn2023.0.2                 amazonlinux                84 k
 pigz                          x86_64               2.5-1.amzn2023.0.3                   amazonlinux                83 k
 runc                          x86_64               1.1.13-1.amzn2023.0.1                amazonlinux               3.2 M

Transaction Summary
=====================================================================================================================================
Install  10 Packages

Total download size: 84 M
Installed size: 317 M
```

Next, we are supposed to configure cgroup in a daemon.json file.
Run the following commands

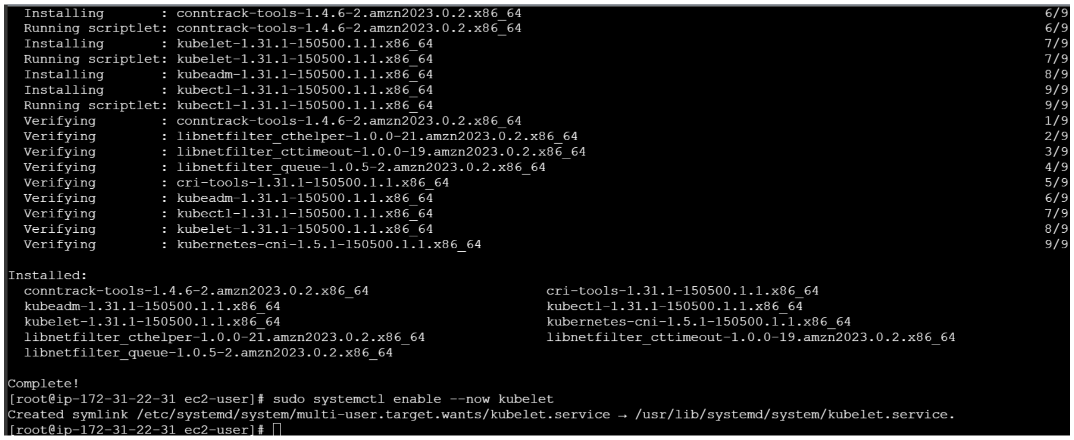
```
cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart
docker
```

Step 4: Kubernetes Installation

Carry out this step on all 3 instances



```
[root@ip-172-31-22-31 ec2-user]# # Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@ip-172-31-22-31 ec2-user]# # This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-22-31 ec2-user]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                                                              48 kB/s | 9.4 kB     00:00
Dependencies resolved.
========================================================================================================================
 Package                              Architecture         Version                              Repository         Size
========================================================================================================================
Installing:
 kubeadm                              x86_64               1.31.1-150500.1.1                    kubernetes         11 M
```

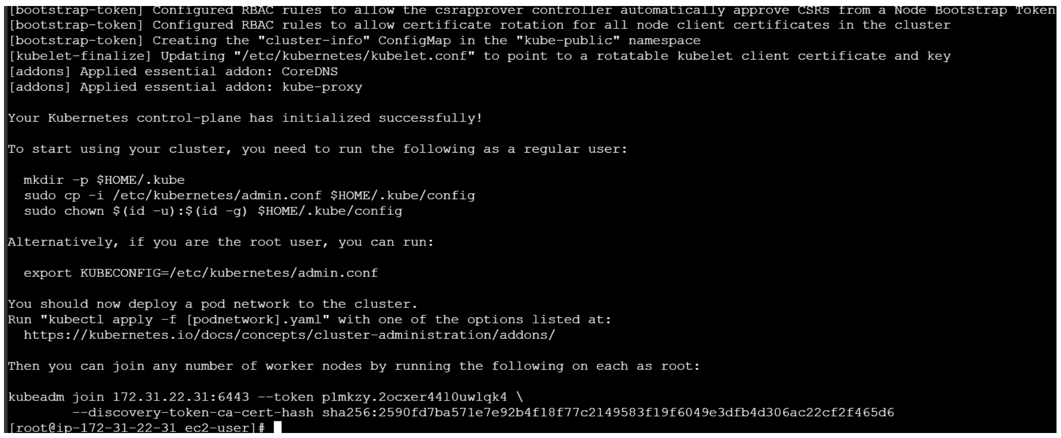


```
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                      6/9
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                      6/9
  Installing       : kubelet-1.31.1-150500.1.1.x86_64                                                                 7/9
  Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64                                                                 7/9
  Installing       : kubeadm-1.31.1-150500.1.1.x86_64                                                                 8/9
  Installing       : kubectl-1.31.1-150500.1.1.x86_64                                                                 9/9
  Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64                                                                 9/9
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                      1/9
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                               2/9
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                              3/9
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                                   4/9
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64                                                               5/9
  Verifying        : kubeadm-1.31.1-150500.1.1.x86_64                                                                 6/9
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64                                                                 7/9
  Verifying        : kubelet-1.31.1-150500.1.1.x86_64                                                                 8/9
  Verifying        : kubernetes-cni-1.5.1-150500.1.1.x86_64                                                           9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                      cri-tools-1.31.1-150500.1.1.x86_64
  kubeadm-1.31.1-150500.1.1.x86_64                                 kubectl-1.31.1-150500.1.1.x86_64
  kubelet-1.31.1-150500.1.1.x86_64                                 kubernetes-cni-1.5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64               libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-22-31 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-22-31 ec2-user]# 
```

After installing Kubernetes,we need to configure internet options to allow bridging.

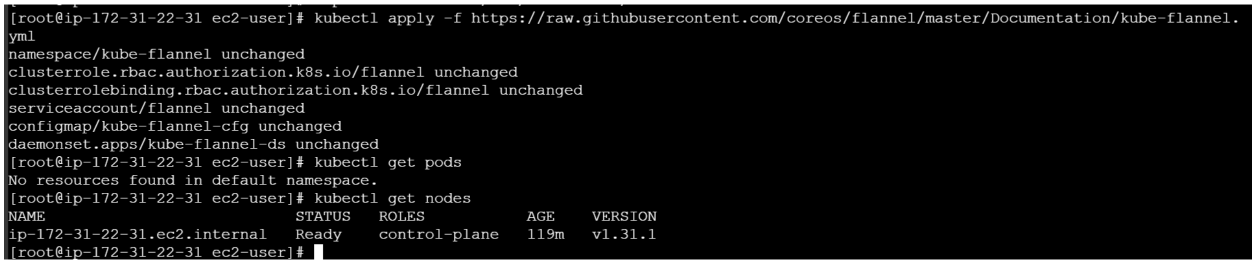
- Sudo swapoff-a
- echo"net.bridge.bridge-nf-call-iptables=1"|sudotee-a/etc/sysctl.conf
- Sudo sysctl-p

Step 5:
On master machine
Run command …kubeadm init with the proper network pod, here it is, --pod-network-cidr=10.244.0.0/16 to initialize kubernetes



```
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.22.31:6443 --token plmkzy.2ocxer4410uwlqk4 \
        --discovery-token-ca-cert-hash sha256:2590fd7ba571e7e92b4f18f77c2149583f19f6049e3dfb4d306ac22cf2f465d6
[root@ip-172-31-22-31 ec2-user]#
```

We are supposed to add a networking plugin named flaggen with the help of the command mentioned in the console output
i.e
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



```
[root@ip-172-31-22-31 ec2-user]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.
yml
namespace/kube-flannel unchanged
clusterrole.rbac.authorization.k8s.io/flannel unchanged
clusterrolebinding.rbac.authorization.k8s.io/flannel unchanged
serviceaccount/flannel unchanged
configmap/kube-flannel-cfg unchanged
daemonset.apps/kube-flannel-ds unchanged
[root@ip-172-31-22-31 ec2-user]# kubectl get pods
No resources found in default namespace.
[root@ip-172-31-22-31 ec2-user]# kubectl get nodes
NAME                          STATUS   ROLES           AGE    VERSION
ip-172-31-22-31.ec2.internal  Ready    control-plane   119m   v1.31.1
[root@ip-172-31-22-31 ec2-user]#
```

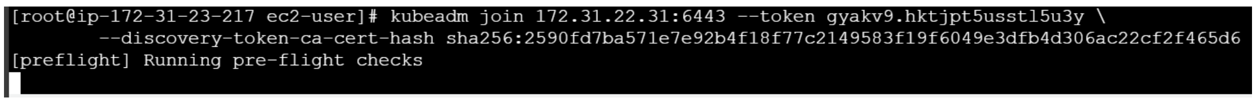
By running kubectl get nodes, we get to see the nodes that are currently connected to the master node

On worker machines
Run the following commands to ensure a smooth and secure joining to the master node

Paste the below command on all 2 worker machines
- sudo yum install iproute-tc-y
- sudo systemctl enable kubelet
- sudo systemctl restart kubelet

Then we are supposed to run the join command that was generated in the console output of our master machine

**kubeadm join 172.31.22.31:6443 --token gyakv9.hktjpt5usstl5u3y \
--discovery-token-ca-cert-hash
sha256:2590fd7ba571e7e92b4f18f77c2149583f19f6049e3dfb4d306ac22cf2f465d6**



```
[root@ip-172-31-23-217 ec2-user]# kubeadm join 172.31.22.31:6443 --token gyakv9.hktjpt5usstl5u3y \
        --discovery-token-ca-cert-hash sha256:2590fd7ba571e7e92b4f18f77c2149583f19f6049e3dfb4d306ac22cf2f465d6
[preflight] Running pre-flight checks
```

Post which we are supposed to get the output that our worker nodes have been successfully connected to master node.

Unfortunately, on running the join command i was not able to produce anything beyond ‘Running pre-filght checks’ which can be seen in the above image

And thus, could not execute the last step of this experiment

Conclusion: In this experiment, we set up a connection between a local machine and an EC2 instance using SSH. After facing issues like timeouts and permission problems, we learned how to check for common causes such as incorrect security group settings, improper key permissions, and network issues. By resolving these, we successfully connected to the EC2 instance. This experiment helped us understand the steps required for remote server access and troubleshooting.